



| | | |
|--|--|---|
|  | <p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 1</p> |

INFORME DE LABORATORIO

(formato estudiante)

| INFORMACIÓN BÁSICA | | | | | |
|---|------------------------------------|----------------------|----------|----------------|----|
| ASIGNATURA: | Fundamentos de la programación 02 | | | | |
| TÍTULO DE LA PRÁCTICA: | Arreglos bidimensionales a objetos | | | | |
| NÚMERO DE PRÁCTICA: | 05 | AÑO LECTIVO: | 2024-B | NRO. SEMESTRE: | // |
| FECHA DE PRESENTACIÓN | 18/10/2024 | HORA DE PRESENTACIÓN | 18:00:00 | | |
| INTEGRANTE (s) Mauro Snayder Sullca Mamani | | | | NOTA (0-20) | |
| DOCENTE(s): Ing. Lino Jose Pinto Oppe | | | | | |

| RESULTADOS Y PRUEBAS | |
|--|--|
| <p>I. EJERCICIOS RESUELTOS:</p> <pre> 7 /** 8 * 9 * @author Mauro Snayder 10 */ 11 public class Soldado { 12 // Creamos los atributos 13 private String nombre="_____"; 14 private int vida; 15 private int fila; 16 private int columna; 17 18 // Creamos los Set y Get de cada atributo 19 public String getNombre() { 20 return nombre; 21 } 22 23 public void setNombre(String nombre) { 24 this.nombre = nombre; 25 } 26 27 public int getVida() { 28 return vida; 29 } 30 31 public int getFila() { 32 return fila; 33 } 34 35 public int getColumna() { 36 return columna; 37 } </pre> | |



```
38 //Generamos una posicion aleatoria para el soldado
39 public void aleatorioPosicion(int fila,int columna){
40     this.fila=(int) (Math.random()*fila);
41     this.columna=(int) (Math.random()*columna);
42 }
43 //Generamos la vida del soldado
44 public void aleatorioVida(){
45     this.vida=(int) (Math.random()*5+1);
46 }
47 // Creamos el toString
48 public String toString() {
49     return "Soldado{" + "nombre=" + nombre + ", vida=" + vida + ", fila=" + fila + ", columna=" + columna + "}\n";
50 }
51 }
```

```
7 /**
8  *
9  * @author Usuario24B
10 */
11
12 public class Actividad05 {
13     public static void main(String[] args) {
14         // Inicializamos dos ejércitos con un número aleatorio de soldados entre 1 y 5
15         int fila=(int) (Math.random()*5+1);//Generamos la fila de la tabla
16         int columna=(int) (Math.random()*5+1);//Generamos la columna de la tabla
17         int numSoldadosPrevio=(int) (Math.random()*10+1);//Generamos la cantidad de soldados
18         //Tomamos el valor menor entre la cantidad de soldados y el numero de espacios que tiene la tabla
19         //ya que puede ocurrir que la cantidad de soldados sea mayor a los espacios de la tabla, lo cual
20         //daria un error
21         int numSoldados=Math.min(numSoldadosPrevio,fila*columna);
22         Soldado[][] ejercito=new Soldado[fila][columna];
23         for (int i=0;i<ejercito.length;i++){//inicializamos el arreglo con soldados sin datos
24             for (int j=0;j<ejercito[i].length;j++){
25                 ejercito[i][j]=new Soldado();
26             }
27         }
28         inicializarEjercito(ejercito,numSoldados);
29         mostrarEjercitoTabla(ejercito);
30         System.out.println("\nEl soldado con mayor vida es: "+mayorVida(ejercito).toString());
31         System.out.println("El promedio de la vida del ejercito es: "+vidaTotal(ejercito)/numSoldados);
32         System.out.println("La vida total del ejercito es: "+vidaTotal(ejercito));
33         System.out.println("\nLista de los soldados por orden de creacion: ");
34         mostrarEjercitoOrdenCreacion(ejercito,numSoldados);
35         System.out.println("\nEl ranking de los soldados es: ");
36         rankingSoldados(ejercito,numSoldados);
37     }
38     // Método para inicializar una tabla con cierto numeros de soldados
39     public static void inicializarEjercito(Soldado[][] ejercito,int numSoldados){
40         for (int i=0;i<numSoldados;i++){
41             Soldado persona=new Soldado();//creamos "persona" para luego ponerlo dentro del arreglo
42             persona.setNombre("soldado"+i);
43             do {
44                 persona.aleatorioPosicion(ejercito.length,ejercito[0].length);//generamos una posicion
45                 persona.aleatorioVida();//generamos la vida
46             }
47             //El bucle se repite si en una posicion aleatoria ya existe un soldado puesto
48             while(!ejercito[persona.getFila()][persona.getColumna()].getNombre().equals("_____"));
49             ejercito[persona.getFila()][persona.getColumna()]=persona;//ponemos el soldado dentro del arreglo
50         }
51     }
```

```

52 // Metodo para mostrar la tabla
53 public static void mostrarEjercitoTabla(Soldado[][] ejercito){
54     for (int i=0;i<ejercito.length;i++){
55         System.out.print("|");
56         for (int j=0;j<ejercito[i].length;j++){
57             System.out.print(ejercito[i][j].getNombre()+"|");
58             System.out.println();
59         }
60     }
61 //Metodo para determinar el soldado con mayor vida
62 public static Soldado mayorVida(Soldado[][] ejercito){
63     Soldado mayor=new Soldado(); //creamos un objeto para almacenar al soldado mayor
64     for (int i=0;i<ejercito.length;i++){
65         for (int j=0;j<ejercito[i].length;j++){
66             if (ejercito[i][j].getVida()>mayor.getVida())
67                 mayor=ejercito[i][j]; //actualizamos "mayor" si otro soldado tiene mayor vida
68         }
69     }
70     return mayor;
71 }
72 //Metodo para determinar la vida total de todos los soldados
73 public static double vidaTotal(Soldado[][] ejercito){
74     double vidaT=0; //vida inicial
75     for (int i=0;i<ejercito.length;i++){
76         for (int j=0;j<ejercito[i].length;j++){
77             vidaT+=ejercito[i][j].getVida(); //sumamos la vida de cada soldado
78         }
79     }
80     return vidaT;
81 }
82 //Metodo para ver las lista de los soldados por el orden de creacion
83 public static void mostrarEjercitoOrdenCreacion(Soldado[][] ejercito,int numSoldados){
84     Soldado[] list=new Soldado[numSoldados]; //creamos un arreglo unidimensional.
85     int copiar=0;
86     for (int j=0;j<ejercito.length;j++){ //Copiamos el arreglo bidimensional a un unidimensional
87         for (int k=0;k<ejercito[j].length;k++){ //ya que es mas facil ordenarlos
88             if (!ejercito[j][k].getNombre().equals(" ")) {
89                 list[copiar]=ejercito[j][k];
90                 copiar++;
91             }
92         }
93     }
94     ordenamientoBurbuja(list); //ordenamos el arreglo unidimensional
95     for (int i=0;i<list.length;i++){ //imprimimos el arreglo
96         System.out.print(list[i].toString());
97     }
98 }
99 //Metodo para ver el ranking de los soldados (por vida)
100 public static void rankingSoldados(Soldado[][] ejercito,int numSoldados){
101     Soldado[] listRankT=new Soldado[numSoldados]; //creamos un arreglo unidimensional.
102     int copiar=0;
103     for (int j=0;j<ejercito.length;j++){ //Copiamos el arreglo bidimensional a un unidimensional
104         for (int k=0;k<ejercito[j].length;k++){ //ya que es mas facil ordenarlos
105             if (!ejercito[j][k].getNombre().equals(" ")) {
106                 listRankT[copiar]=ejercito[j][k];
107                 copiar++;
108             }
109         }
110     }
111     ordenamientoInsercion(listRankT); //ordenamos el arreglo unidimensional
112     for (int i=0,j=listRankT.length-1;j>=0;i++,j--){ //imprimimos el arreglo
113         System.out.print((i+1)+" -> "+listRankT[j].toString());
114     }

```

| | | |
|---|---|---|
|  | <p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 4</p> |

```

115 //Metodo de ordenamiento burbuja para los nombres de los soldados, ya que de esa
116 //manera podemos ver el orden de creacion de los soldados
117 public static void ordenamientoBurbuja(Soldado[] lista){
118     Soldado cambio;
119     for(int i=0;i<lista.length-1;i++){
120         for(int j=0;j<lista.length-i-1;j++){
121             if(lista[j].getNombre().compareTo(lista[j+1].getNombre())>0){
122                 cambio=lista[j];
123                 lista[j]=lista[j+1];
124                 lista[j+1]=cambio;
125             }
126         }
127     }
128 }
129 //Metodo de ordenamiento de insercion para la vida de los soldados
130 public static void ordenamientoInsercion(Soldado[] lista) {
131     for (int i=1;i<lista.length;i++) {
132         Soldado soldadoActual=lista[i];
133         int j=i-1;
134         while (j>=0 && lista[j].getVida()>soldadoActual.getVida()){
135             lista[j+1]=lista[j];
136             j--;
137         }
138         lista[j+1]=soldadoActual;
139     }
140 }
141 }

```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

Se utilizaron valores aleatorios para definir las filas y columnas de la tabla, así como el número de soldados, variando entre 1 y 5 para las dimensiones y entre 1 y 10 para la cantidad de soldados. Esto permitió evaluar cómo el programa manejaba diferentes configuraciones y asegurarse de que los soldados se asignaran correctamente a las posiciones en la tabla.

¿Qué resultado esperabas obtener para cada valor de entrada?

Se esperaba que, dependiendo de los valores de entrada, la tabla se llenara correctamente sin solapamientos de soldados. Además, se anticipaba que el programa identificara el soldado con mayor vida, calculase la vida total y el promedio de vida, y que mostrara la lista de soldados en el orden de creación, así como su ranking por vida.

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

Al ejecutar el programa, se observó que los resultados variaban por la aleatoriedad en las posiciones y vidas de los soldados. La lista de soldados se presentó en el orden de creación, y el ranking se mostró correctamente, indicando que el programa funcionó de acuerdo con lo esperado en general.

```
El ranking de los soldados es:
1 -> Soldado[nombre=soldado0, vida=3, fila=1, columna=0]
2 -> Soldado[nombre=soldado1, vida=3, fila=0, columna=4]
3 -> Soldado[nombre=soldado4, vida=3, fila=0, columna=2]
4 -> Soldado[nombre=soldado3, vida=2, fila=3, columna=2]
5 -> Soldado[nombre=soldado2, vida=1, fila=0, columna=3]

BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - Laboratorios (run)



Lista de los soldados por orden de creacion:

```
Soldado{nome=soldado0, vida=3, fila=2, columna=4}
Soldado{nome=soldado1, vida=4, fila=0, columna=3}
Soldado{nome=soldado2, vida=3, fila=0, columna=0}
Soldado{nome=soldado3, vida=4, fila=0, columna=2}
Soldado{nome=soldado4, vida=4, fila=1, columna=3}
Soldado{nome=soldado5, vida=1, fila=2, columna=0}
Soldado{nome=soldado6, vida=2, fila=2, columna=1}
Soldado{nome=soldado7, vida=4, fila=0, columna=4}
Soldado{nome=soldado8, vida=4, fila=0, columna=1}
```

El ranking de los soldados es:

```
1 -> Soldado{nombre=soldado4, vida=4, fila=1, columna=3}
2 -> Soldado{nombre=soldado7, vida=4, fila=0, columna=4}
3 -> Soldado{nombre=soldado1, vida=4, fila=0, columna=3}
4 -> Soldado{nombre=soldado3, vida=4, fila=0, columna=2}
5 -> Soldado{nombre=soldado8, vida=4, fila=0, columna=1}
6 -> Soldado{nombre=soldado0, vida=3, fila=2, columna=4}
7 -> Soldado{nombre=soldado2, vida=3, fila=0, columna=0}
8 -> Soldado{nombre=soldado6, vida=2, fila=2, columna=1}
9 -> Soldado{nombre=soldado5, vida=1, fila=2, columna=0}
```

```
BUILD SUCCESSFUL (total time: 0 seconds)
```

| | | |
|---|--|---|
|  | <p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 7</p> |

III. COMMITS:

Entramos a nuestra carpeta donde están nuestros archivos y añadimos los cambios.

```

MINGW64:/c/Users/Mauro Snyder/Documents/NetBeansProjects/Laboratorios/src
Mauro Snyder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Laboratorio_05/Actividad05.java
        modified:   Laboratorio_05/Soldado.java

no changes added to commit (use "git add" and/or "git commit -a")

Mauro Snyder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git add .

Mauro Snyder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Laboratorio_05/Actividad05.java
        modified:   Laboratorio_05/Soldado.java

Mauro Snyder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```

Hacemos un commit

```

Mauro Snyder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git commit -m "acabado"
[master 3253f3e] acabado
 2 files changed, 124 insertions(+), 37 deletions(-)

Mauro Snyder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```


Subimos nuestro commit a nuestro repositorio remoto



```
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git push -u origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.26 KiB | 2.26 MiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/MauroSullcaMamani/FDLP2_LAB.git
 9305b36..3253f3e master -> master
branch 'master' set up to track 'origin/master'.
```

Link de mi repositorio: https://github.com/MauroSullcaMamani/FDLP2_LAB.git

IV. RUBRICA:

| Contenido y demostración | | Puntos | Checklis t | Estudiant e | Profeso r |
|--------------------------|--|--------|---------------|----------------|--------------|
| 1. GitHub | Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar. | 2 | ✓ | 2 | |
| 2. Commits | Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación). | 4 | ✓ | 3 | |
| 3. Código fuente | Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones. | 2 | ✓ | 2 | |
| 4. Ejecución | Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente. | 2 | ✓ | 2 | |
| 5. Pregunta | Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación). | 2 | ✓ | 2 | |
| 6. Fechas | Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos. | 2 | ✓ | 2 | |
| 7. Ortografía | El documento no muestra errores ortográficos. | 2 | ✓ | 2 | |
| 8. Madurez | El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación). | 4 | ✓ | 3 | |
| TOTAL | | 20 | | 18 | |

Tabla 2: Rúbrica para contenido del Informe y demostración

| | | |
|---|---|---|
|  | <p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 9</p> |

CONCLUSIONES

En conclusión, la utilización de arreglos bidimensionales de objetos en el programa, junto con técnicas de inicialización, búsqueda y ordenamiento, permite manejar colecciones de soldados de manera eficiente y organizada. Al trabajar con objetos, se pueden gestionar múltiples atributos, como nombre y vida, de forma estructurada, lo que facilita la manipulación y el acceso a la información. Las funcionalidades implementadas para identificar al soldado con mayor vida, calcular la vida total y promedio, y ordenar soldados por su creación y vida, mejoran la accesibilidad y optimizan el rendimiento del programa, proporcionando una base sólida para futuras mejoras.

METODOLOGÍA DE TRABAJO

Lo primero que hice, es leer cada los enunciados de cada ejercicio y también tomar en cuenta las restricciones que nos da para así poder buscar una solución al problema. Después observar el código y entender la funcionalidad de cada uno y completar las partes que están incompletas. Y por último comprobar nuestro código ingresando varias veces valores de prueba para ver que nuestro código está funcionando correctamente.

REFERENCIAS Y BIBLIOGRAFÍA

E. G. Castro Gutiérrez and M. W. Aedo López, Fundamentos de programación 2: tópicos de programación orientada a objetos, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021, pp. 170, ISBN 978-612 5035-20-2.